**Lab 08**

**Name:** Will Townsend

**Class:** COSC 320 – Section 751

**Date:** 11/12/2020

**Lab Report:** I can tell this lab was made to be a simpler one (only took about a half an hour including this report), but the mechanics behind it are more complicated. I think I have a clear understanding of heaps and priority queues after the discussions in class and doing this lab. I did this lab 100% independently with no help from outside sources (i.e. classmates or internet).

**\*\*\*NO WRITTEN PRELAB\*\*\***

**Lab Tasks:**

Task 1:

#ifndef PROCESSREQUESTRECORD\_CLASS

#define PROCESSREQUESTRECORD\_CLASS

#include <iostream>

#include <string>

using namespace std;

class procReqRec

{

public:

// default constructor

procReqRec()

{}

// constructor

procReqRec(const string& nm, int p){

name=nm;

priority=p;

}

// access functions

int getPriority(){return priority;}

string getName(){return name;}

// update functions

void setPriority(int p){priority=p;}

void setName(const string& nm){name=nm;}

// for maintenance of a minimum priority queue

friend bool operator> (const procReqRec& left,const procReqRec& right){

if(left.priority<right.priority)

return true;

return false;

}

// output a process request record in the format

// name: priority

friend ostream& operator<< (ostream& ostr,const procReqRec& obj){

ostr<<obj.name<<"("<<obj.priority<<")\n";

return ostr;

}

private:

string name; // process name

int priority; // process priority

};

#endif // PROCESSREQUESTRECORD\_CLASS

Task 3:

#include<stdio.h>

#include<iostream>

#include<time.h>

#include<stdlib.h>

#include"preqrec.h"

#include"d\_pqueue.h"

int main(){

miniPQ<procReqRec> mpq;

srand(time(0));

mpq.push(procReqRec("Process A",rand()%40));

mpq.push(procReqRec("Process B",rand()%40));

mpq.push(procReqRec("Process C",rand()%40));

mpq.push(procReqRec("Process D",rand()%40));

mpq.push(procReqRec("Process E",rand()%40));

mpq.push(procReqRec("Process F",rand()%40));

mpq.push(procReqRec("Process G",rand()%40));

mpq.push(procReqRec("Process H",rand()%40));

mpq.push(procReqRec("Process I",rand()%40));

mpq.push(procReqRec("Process J",rand()%40));

puts("Lab 08:");

while(mpq.size()>0){

std::cout<<mpq.top();

mpq.pop();

}

return 0;

}

Output 1:

Lab 08:

Process J(6)

Process F(22)

Process H(23)

Process B(26)

Process A(28)

Process I(29)

Process C(32)

Process G(34)

Process E(34)

Process D(39)

Output 2:

Lab 08:

Process C(2)

Process A(6)

Process H(10)

Process E(12)

Process B(14)

Process I(20)

Process D(20)

Process F(21)

Process J(36)

Process G(37)

Output 3:

Lab 08:

Process C(5)

Process I(6)

Process J(8)

Process G(9)

Process H(13)

Process E(17)

Process F(23)

Process A(25)

Process D(25)

Process B(30)

Output 4:

Lab 08:

Process G(1)

Process E(8)

Process F(9)

Process C(14)

Process I(24)

Process J(26)

Process D(29)

Process B(32)

Process A(33)

Process H(37)

Output 5:

Lab 08:

Process A(0)

Process J(4)

Process H(9)

Process B(11)

Process G(13)

Process F(25)

Process I(27)

Process E(31)

Process D(37)

Process C(38)